SmartCat

Nina Marjanovic

REINFORCEMENT LEARNING

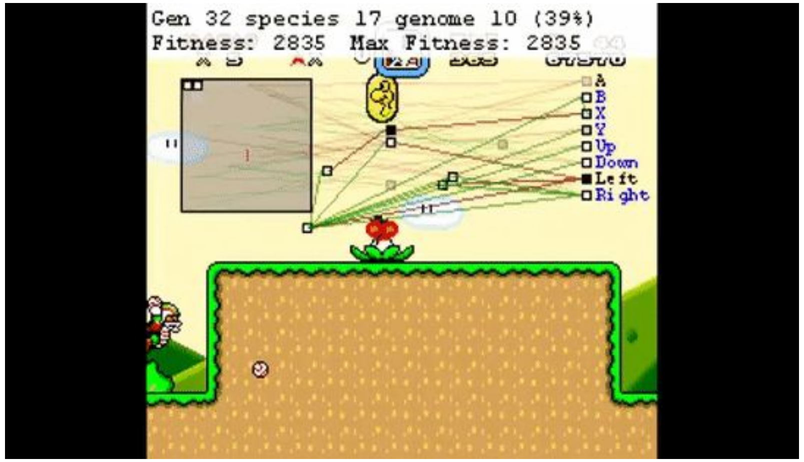
DATA ⟶ KNOWLEDGE ⟶ POWER

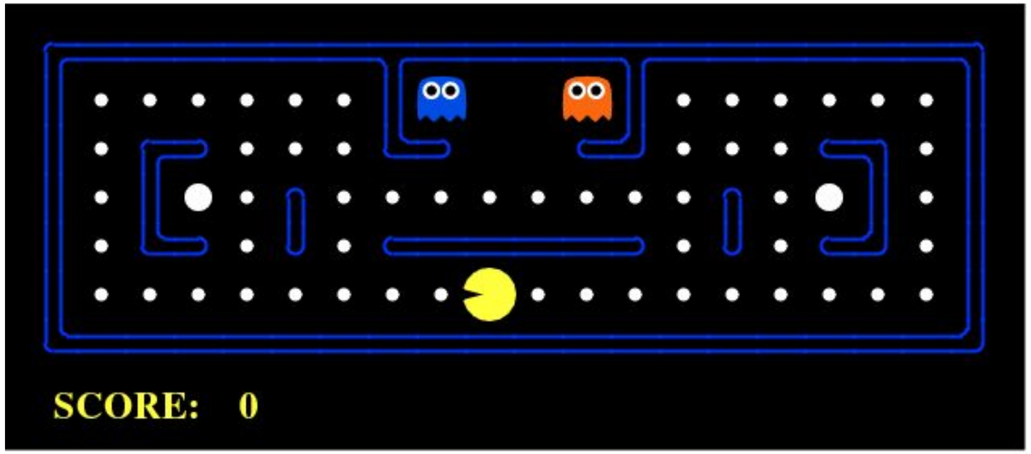
Lecture outline

- About reinforcement learning (~30 minutes)
    - Definition of reinforcement learning
    - Difference between supervised learning and reinforcement learning
    - Markov decision process (definition and problem specification)
    - Exploration and exploitation
    - Environments and agents
- RL Algorithms (~15 minutes)
    - What is a policy?
    - Q-learning
- Workshop - working on RL problems using Q-learning(~45 minutes)
    - The pole balancing problem
    - Driving up a big hill
- Advanced algorithms (~15 minutes)
    - Types of RL algorithms
    - NEAT
    - DDPG
- Workshop - working on RL problem using DDPG (~30 minutes)
    - Pendulum

Motivation

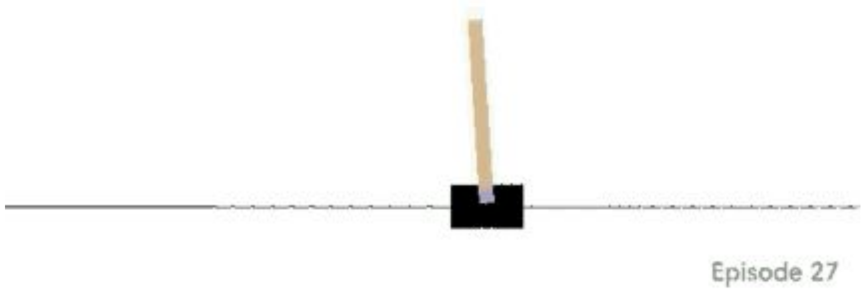

ima...
019 5 1

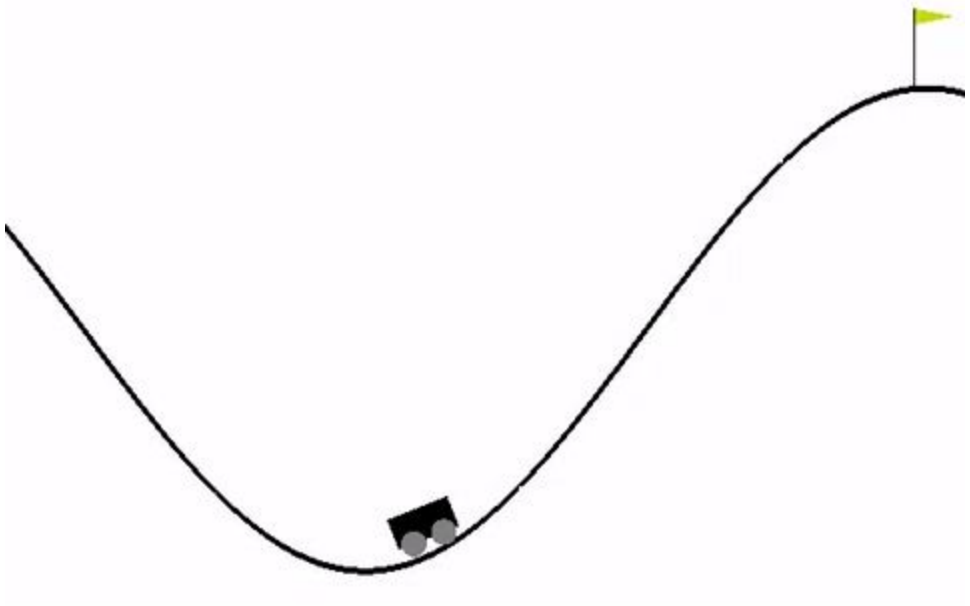

Gen 32 species 17 genome 10 (39%)
Fitness: 2835 Max Fitness: 2835
A
B
X
Y
Up
Down
Left
Right


SCORE: 0

SmartCat

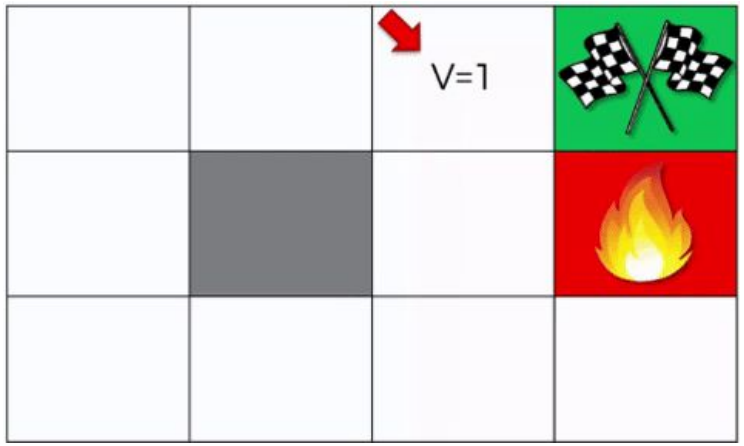
Motivation



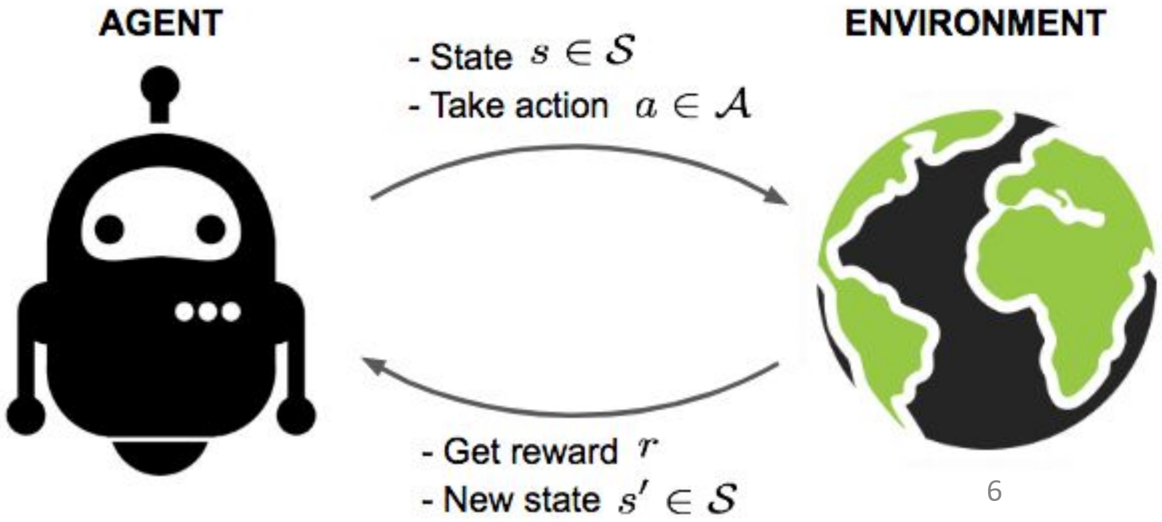


About reinforcement learning

Definition of reinforcement learning



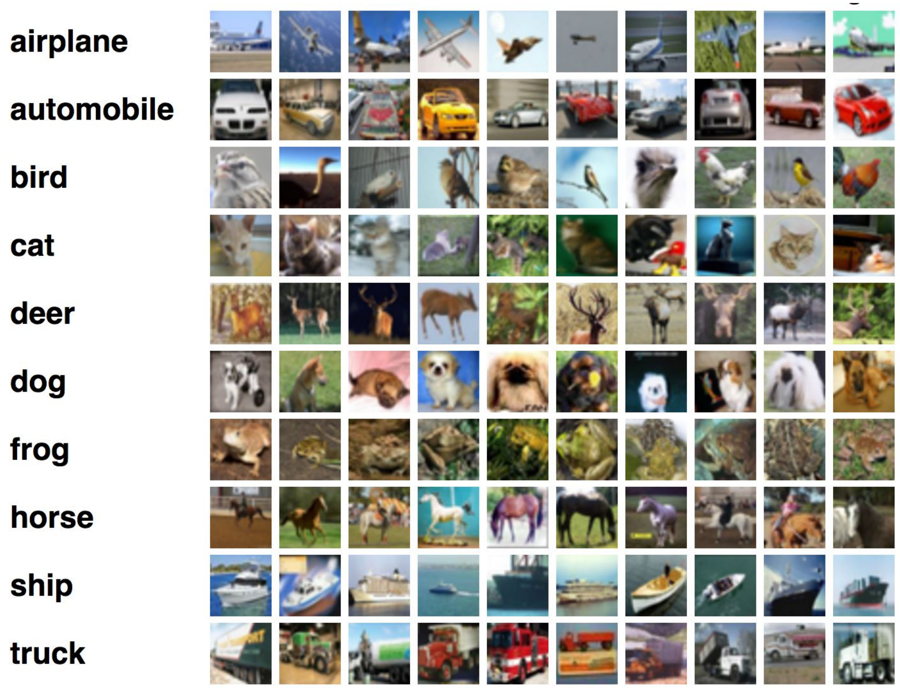
- Type of machine learning
- Software agents and machines automatically learn how to behave
- Reward feedback



SmartCat

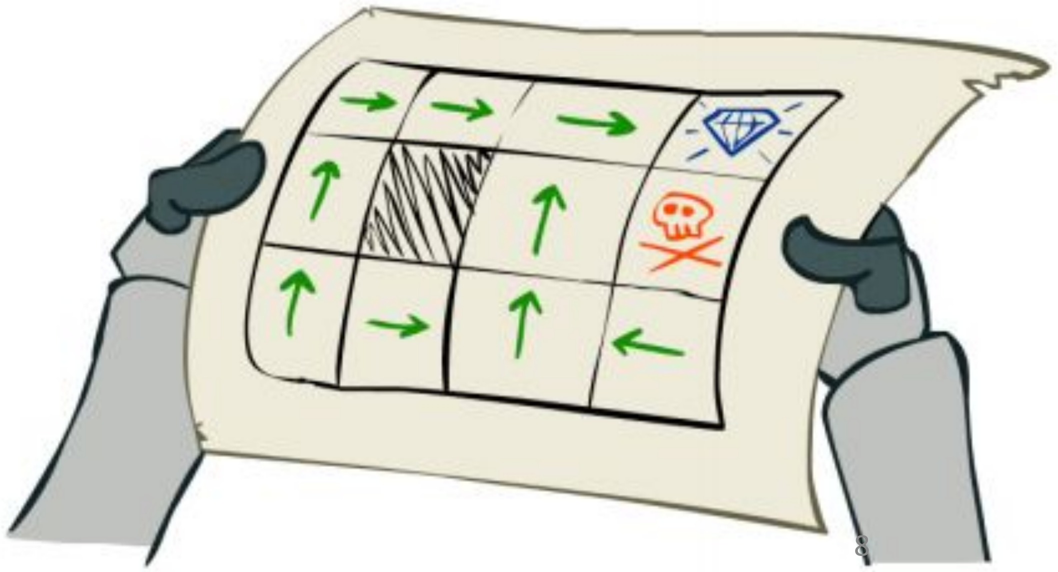
Difference between supervised and reinforcement learning

- Why do these two paradigms co-exist?
- What are the downsides of using a supervised learning method to solve a RL problem?



Markov decision process

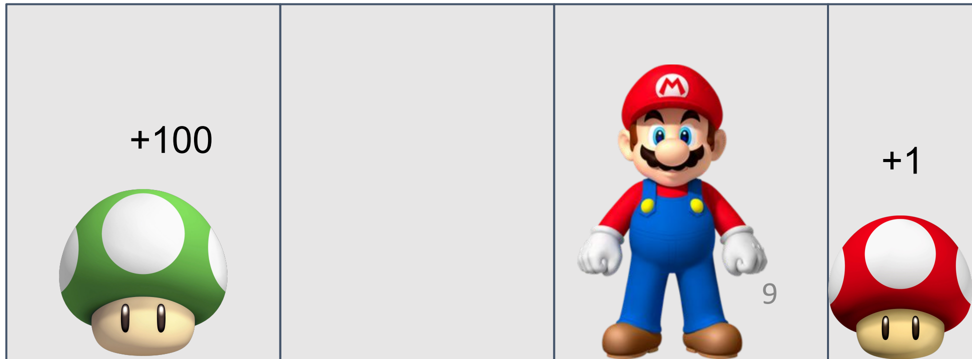
- Definition
- (S, A, P_a, R_a, γ)
  - S - finite set of states
  - A - finite set of actions
  - $P_a(s, s') = Pr(s_{t+1} = s' \mid s_t = s, a_t = a)$ - state transition model
  - $R_a(s, s')$ - reward model
  - $\gamma \in [0, 1]$ - discount factor
- Finding a policy
- Discrete time stochastic control process - rewards and probabilities
- How does reinforcement learning relate to MDPs?



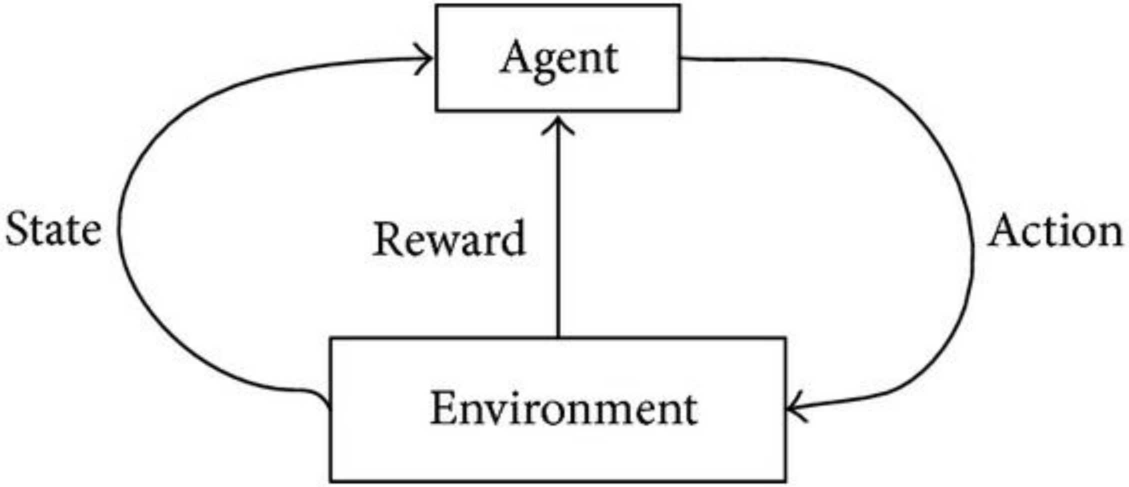
Exploration vs. Exploitation

- Explore - collect more information
- Exploit - use the current information to make the best decision

- ε - greedy action



Environments and agents

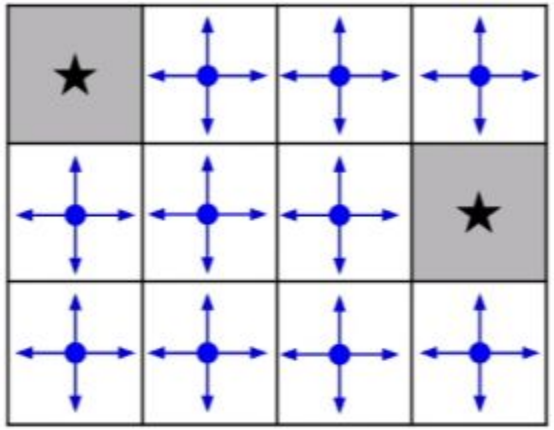


SmartCat

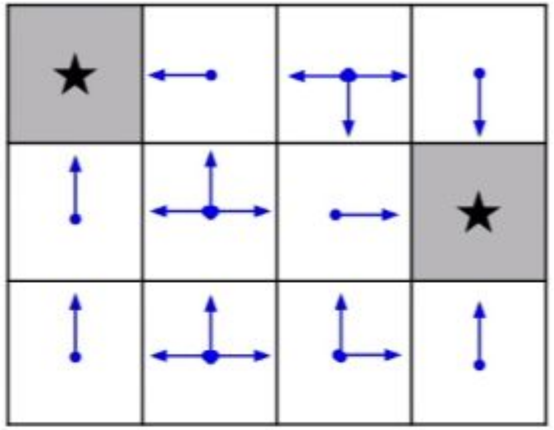
Algorithms

SmartCat

What is a policy?



Random Policy



Optimal Policy

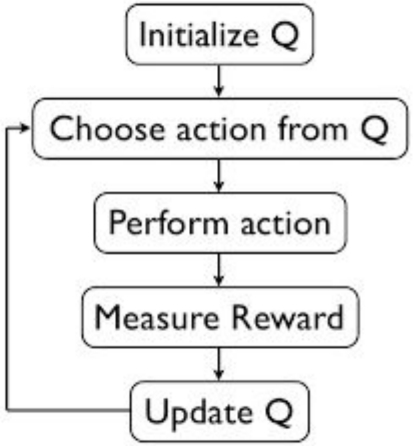
SmartCat

Q - Learning

- The Bellman Equation
  $Q(s, a) = r + \gamma \max_{a'} Q(s', a')$
  - r - immediate reward
  - γ - discount factor
  - $\gamma \max_{a'} Q(s', a')$ - future reward

$$Q = \begin{array}{c c} & \begin{array}{cccccc} 0 & 1 & 2 & 3 & 4 & 5 \end{array} \\ \begin{array}{c} 0 \\ 1 \\ 2 \\ 3 \\ 4 \\ 5 \end{array} & \begin{bmatrix} 0 & 0 & 0 & 0 & 400 & 0 \\ 0 & 0 & 0 & 320 & 0 & 500 \\ 0 & 0 & 0 & 320 & 0 & 0 \\ 0 & 400 & 256 & 0 & 400 & 0 \\ 320 & 0 & 0 & 320 & 0 & 500 \\ 0 & 400 & 0 & 0 & 400 & 500 \end{bmatrix} \end{array}$$

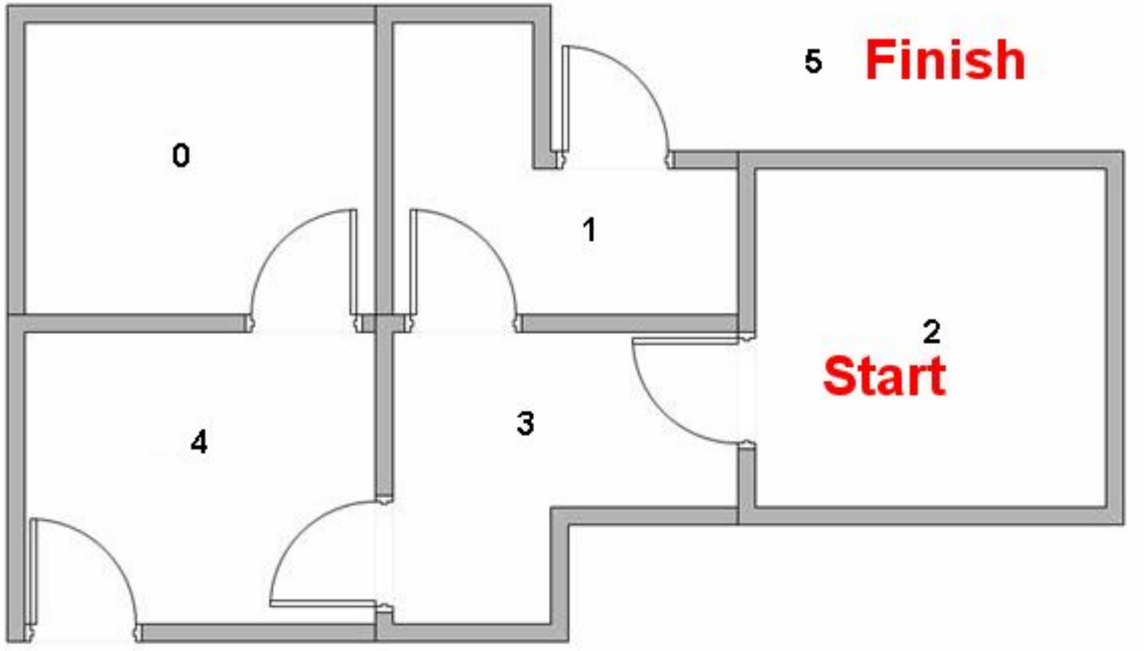
- Algorithm:



Initialize $Q(s,a), \forall s \in \mathcal{S}, a \in \mathcal{A}(s)$, arbitrarily, and $Q(\textit{terminal-state}, \cdot) = 0$
Repeat (for each episode):
  Initialize S
  Choose A from S using policy derived from Q (e.g., ε-greedy)
  Repeat (for each step of episode):
    Take action A, observe R, S'
    Choose A' from S' using policy derived from Q (e.g., ε-greedy)
    $Q(S,A) \leftarrow Q(S,A) + \alpha\left[R + \gamma Q(S',A') - Q(S,A)\right]$
    $S \leftarrow S'$; $A \leftarrow A'$;
  until S is terminal

SmartCat

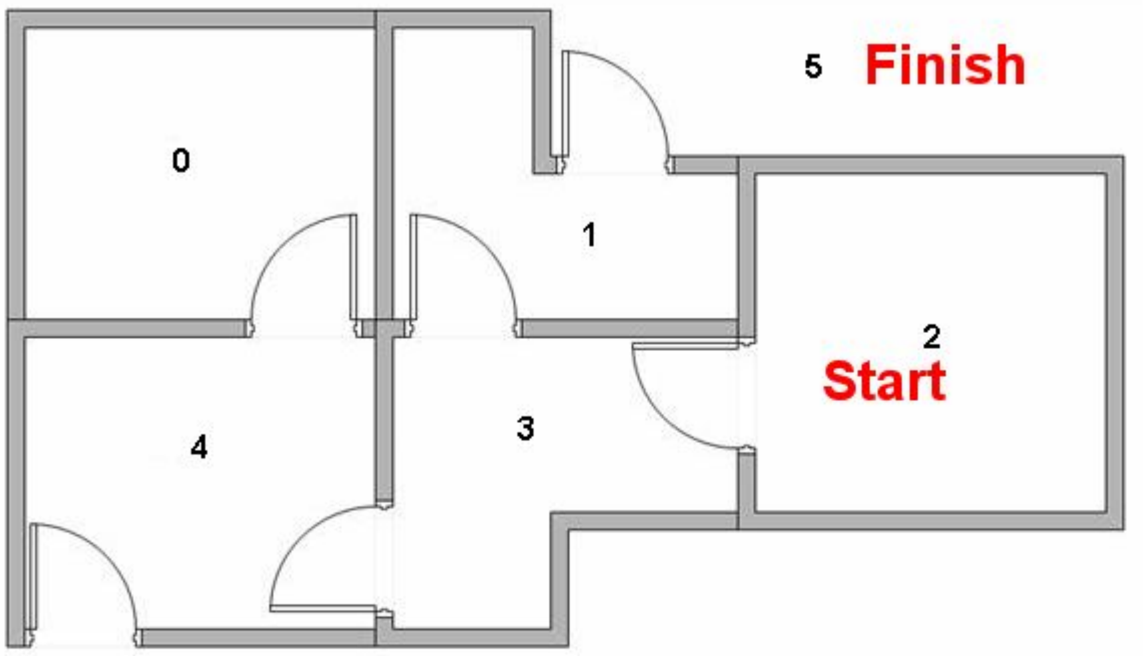
Q - Learning - Example



$$
R = \begin{array}{cc}
 & \text{Action} \\
\text{State} & \begin{array}{cccccc} 0 & 1 & 2 & 3 & 4 & 5 \end{array} \\
\begin{array}{c} 0 \\ 1 \\ 2 \\ 3 \\ 4 \\ 5 \end{array} & \begin{bmatrix} -1 & -1 & -1 & -1 & 0 & -1 \\ -1 & -1 & -1 & 0 & -1 & 100 \\ -1 & -1 & -1 & 0 & -1 & -1 \\ -1 & 0 & 0 & -1 & 0 & -1 \\ 0 & -1 & -1 & 0 & -1 & 100 \\ -1 & 0 & -1 & -1 & 0 & 100 \end{bmatrix}
\end{array}
$$

Q(state, action) = R(state, action) + Gamma * Max[Q(next state, all actions)]

Q - Learning - Example

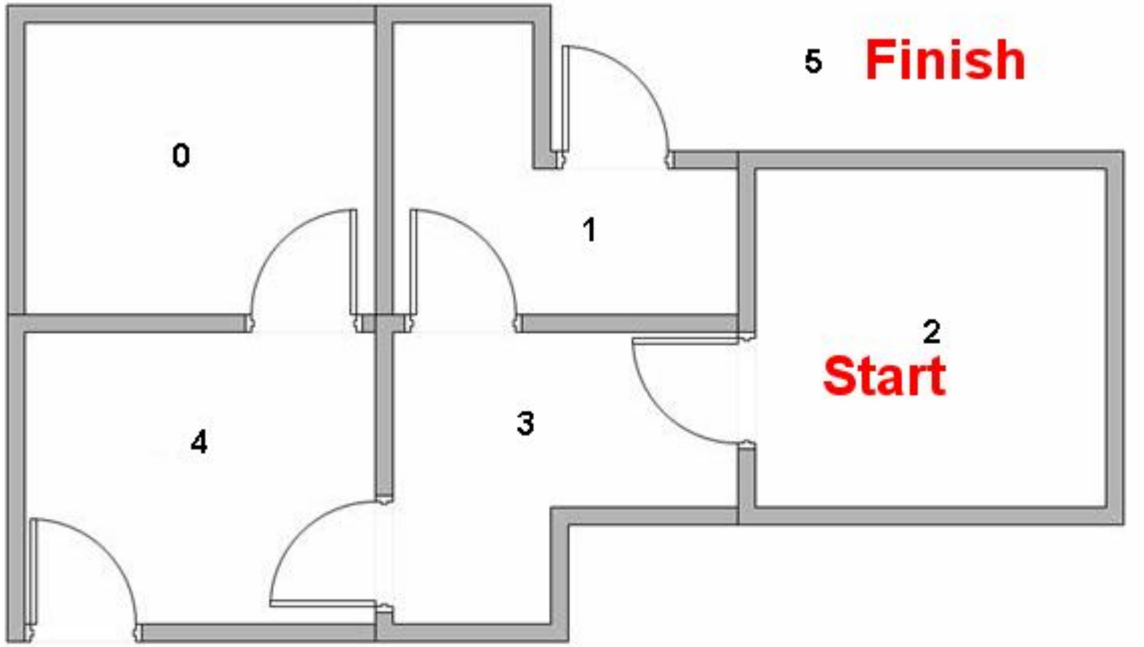


$$
Q = \begin{array}{c c}
 & \begin{array}{cccccc} 0 & 1 & 2 & 3 & 4 & 5 \end{array} \\
\begin{array}{c} 0 \\ 1 \\ 2 \\ 3 \\ 4 \\ 5 \end{array} &
\begin{bmatrix}
0 & 0 & 0 & 0 & 0 & 0 \\
0 & 0 & 0 & 0 & 0 & 0 \\
0 & 0 & 0 & 0 & 0 & 0 \\
0 & 0 & 0 & 0 & 0 & 0 \\
0 & 0 & 0 & 0 & 0 & 0 \\
0 & 0 & 0 & 0 & 0 & 0
\end{bmatrix}
\end{array}
$$

Q(state, action) = R(state, action) + Gamma * Max[Q(next state, all actions)]

SmartCat

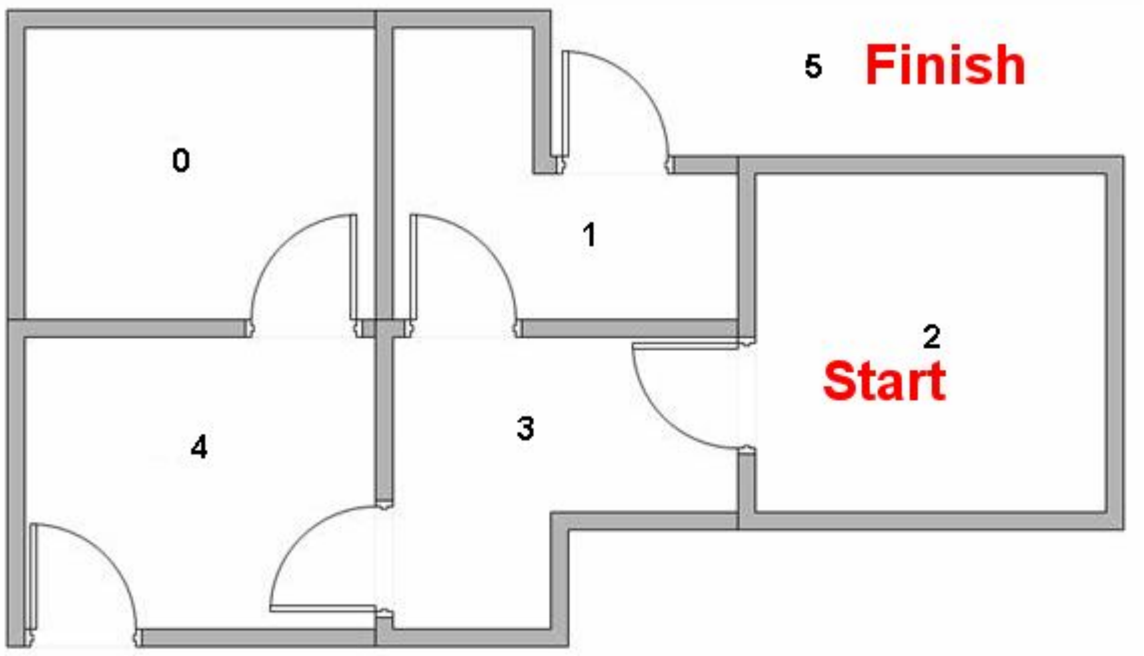
Q - Learning - Example



$$Q = \begin{array}{c c} & \begin{array}{cccccc} 0 & 1 & 2 & 3 & 4 & 5 \end{array} \\ \begin{array}{c} 0 \\ 1 \\ 2 \\ 3 \\ 4 \\ 5 \end{array} & \begin{bmatrix} 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 100 \\ 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 \end{bmatrix} \end{array}$$

Q(state, action) = R(state, action) + Gamma * Max[Q(next state, all actions)]
Q(1, 5) = R(1, 5) + 0.8 * Max[Q(5, 1), Q(5, 4), Q(5, 5)] = 100 + 0.8 * 0 = 100

Q - Learning - Example

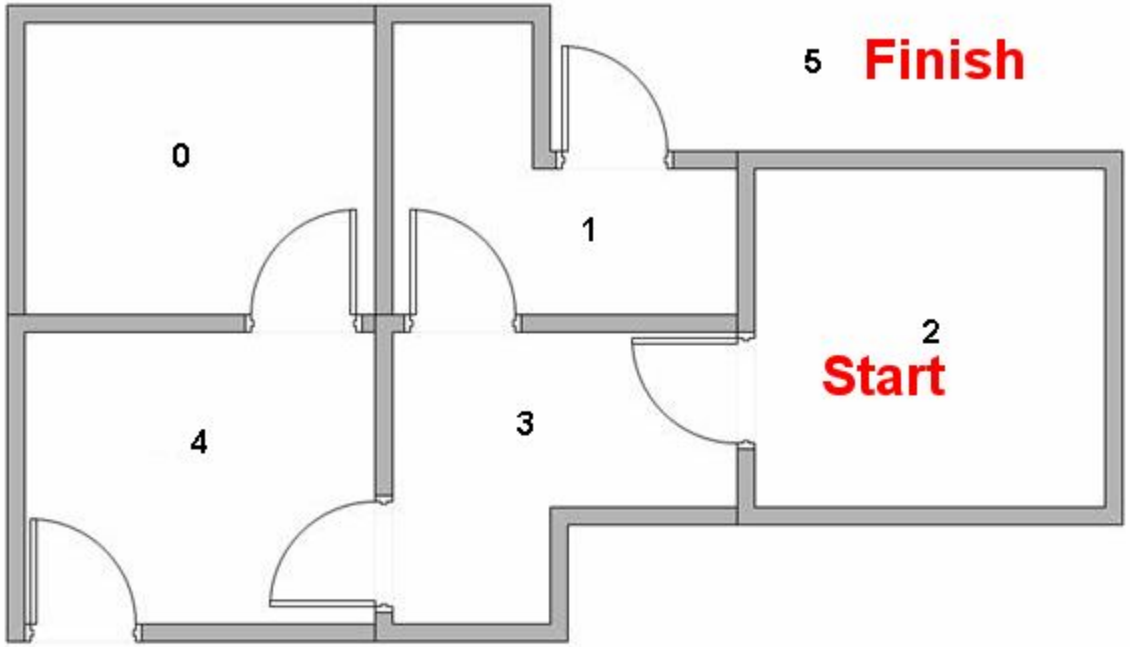


$$Q = \begin{array}{c} \\ 0 \\ 1 \\ 2 \\ 3 \\ 4 \\ 5 \end{array} \begin{array}{c} \begin{array}{cccccc} 0 & 1 & 2 & 3 & 4 & 5 \end{array} \\ \begin{bmatrix} 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 100 \\ 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 80 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 \end{bmatrix} \end{array}$$

Q(state, action) = R(state, action) + Gamma * Max[Q(next state, all actions)]
Q(3, 1) = R(3, 1) + 0.8 * Max[Q(1, 3), Q(1, 5)] = 0 + 0.8 * Max(0, 100) = 80

SmartCat

Q - Learning - Example



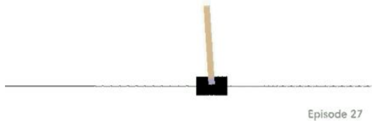
$$
Q = \begin{array}{c} \\ 0 \\ 1 \\ 2 \\ 3 \\ 4 \\ 5 \end{array}
\begin{array}{c}
\begin{array}{cccccc} 0 & 1 & 2 & 3 & 4 & 5 \end{array} \\
\begin{bmatrix}
0 & 0 & 0 & 0 & 80 & 0 \\
0 & 0 & 0 & 64 & 0 & 100 \\
0 & 0 & 0 & 64 & 0 & 0 \\
0 & 80 & 51 & 0 & 80 & 0 \\
64 & 0 & 0 & 64 & 0 & 100 \\
0 & 80 & 0 & 0 & 80 & 100
\end{bmatrix}
\end{array}
$$

Q(state, action) = R(state, action) + Gamma * Max[Q(next state, all actions)]

SmartCat

Workshop

Cart pole



Observation
Type: Box(4)

Num	Observation	Min	Max
0	Cart Position	-2.4	2.4
1	Cart Velocity	-Inf	Inf
2	Pole Angle	~ -41.8°	~ 41.8°
3	Pole Velocity At Tip	-Inf	Inf

Actions
Type: Discrete(2)

Num	Action
0	Push cart to the left
1	Push cart to the right

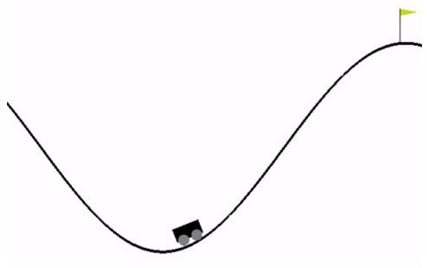
Note: The amount the velocity is reduced or increased is not fixed as it depends on the angle the pole is pointing. This is because the center of gravity of the pole increases the amount of energy needed to move the cart underneath it

Reward
Reward is 1 for every step taken, including the termination step

Starting State
All observations are assigned a uniform random value between ±0.05

Episode Termination
Pole Angle is more than ±12°
Cart Position is more than ±2.4 (center of the cart reaches the edge of the display)
Episode length is greater than 200

Mountain car



Observation

Type: Box(2)

Num	Observation	Min	Max
0	position	-1.2	0.6
1	velocity	-0.07	0.07

Actions

Type: Discrete(3)

Num	Observation
0	push left
1	no push
2	push right

Reward

-1 for each time step, until the goal position of 0.5 is reached. As with MountainCarContinuous v0, there is no penalty for climbing the left hill, which upon reached acts as a wall.

Starting State

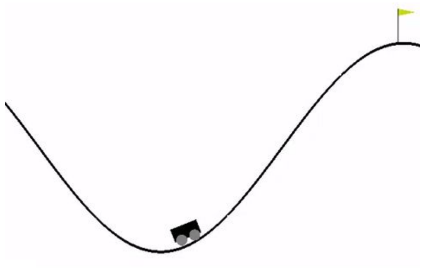
Random position from -0.6 to -0.4 with no velocity.

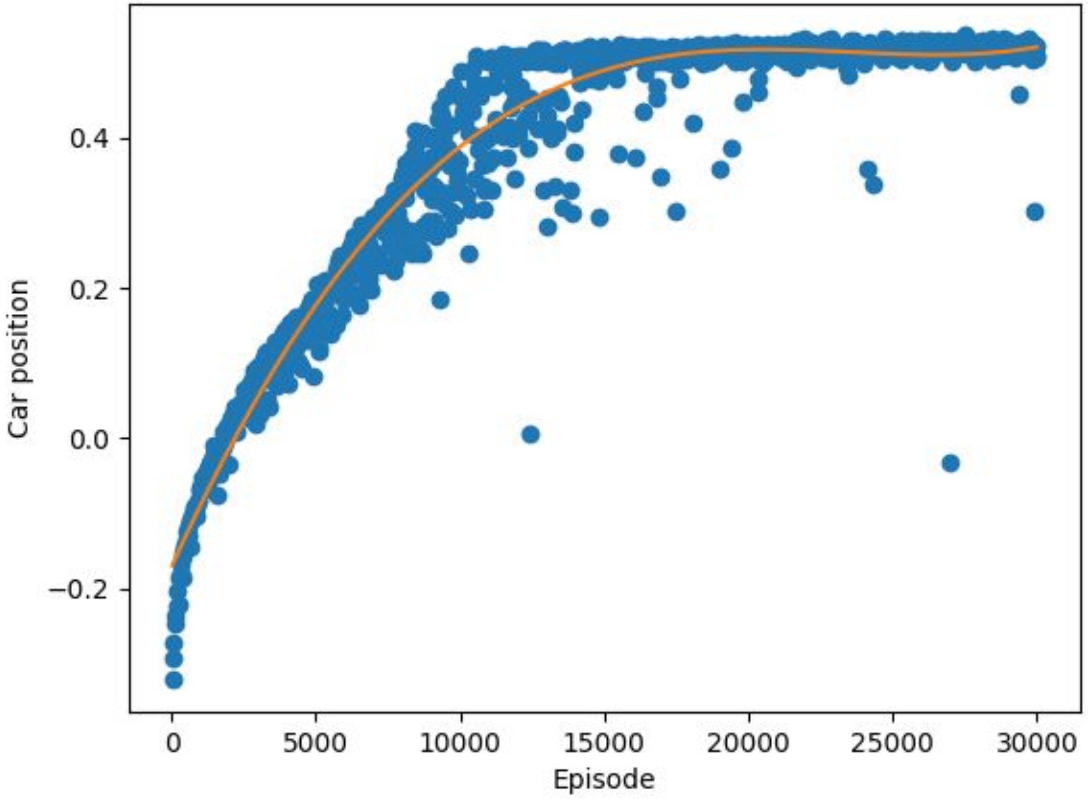
Episode Termination

The episode ends when you reach 0.5 position, or if 200 iterations are reached.

SmartCat

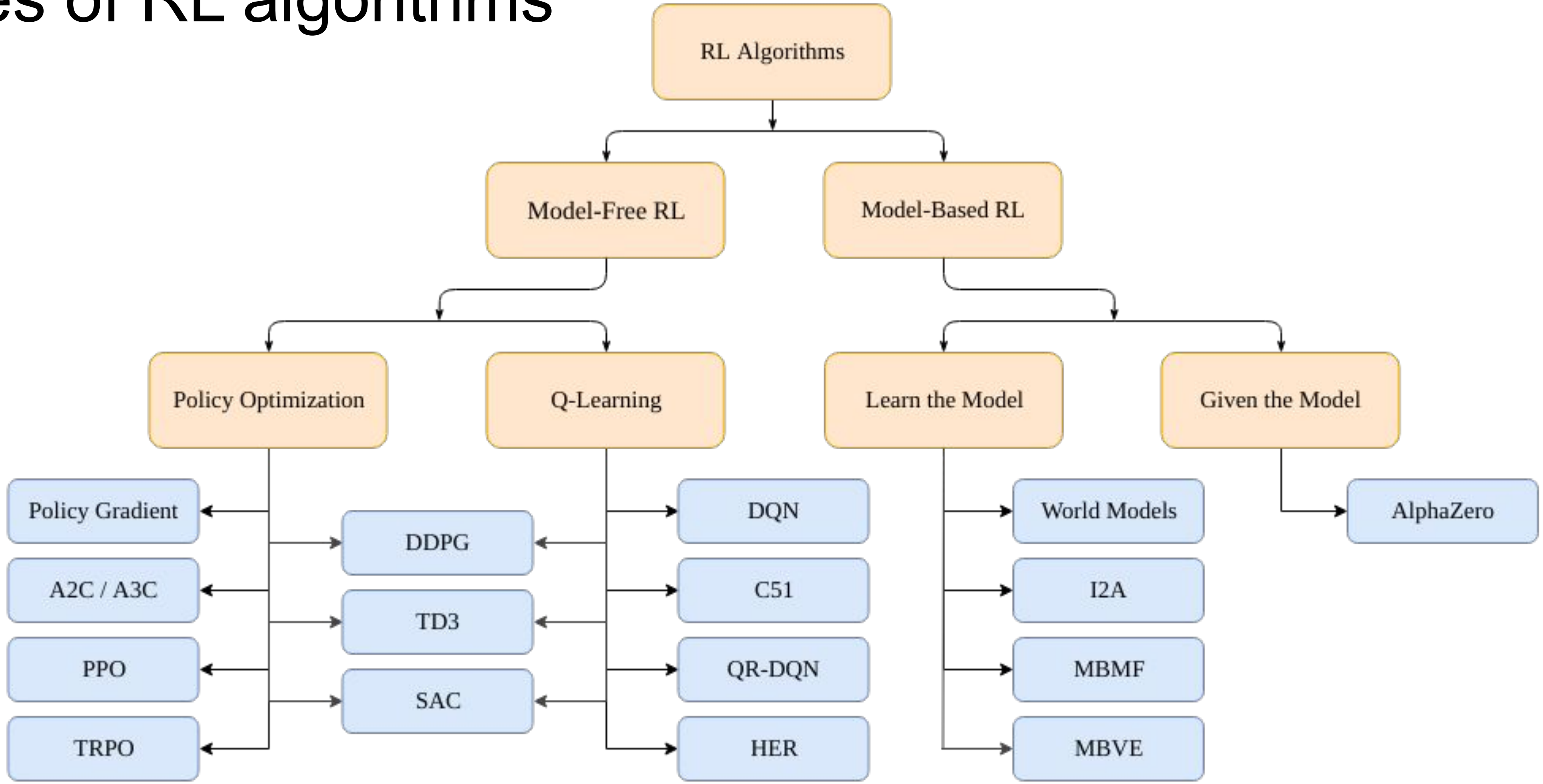
Mountain car





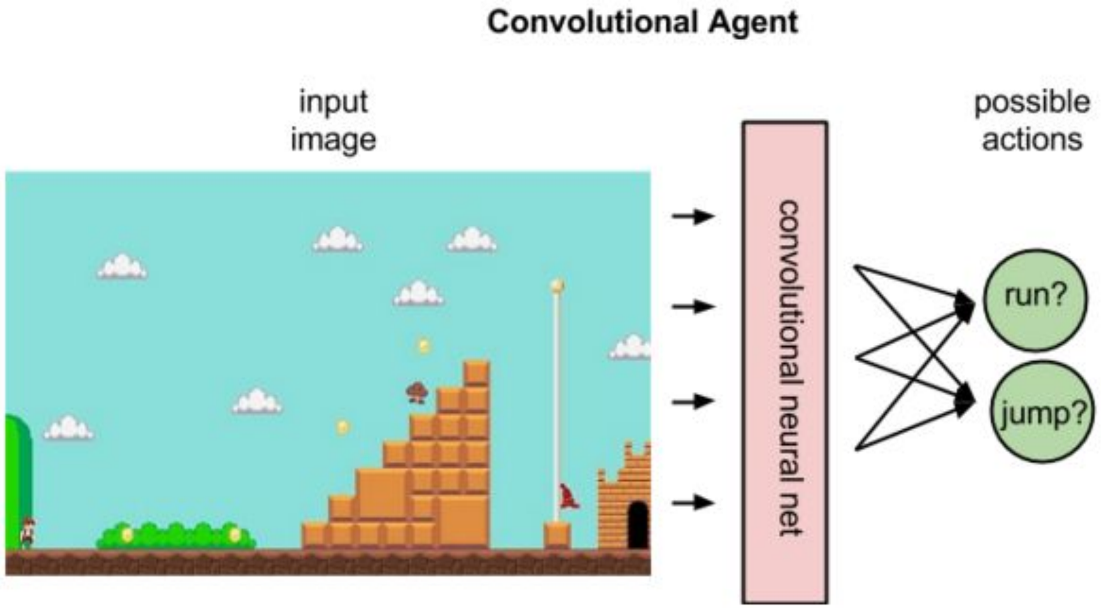
Advanced algorithms

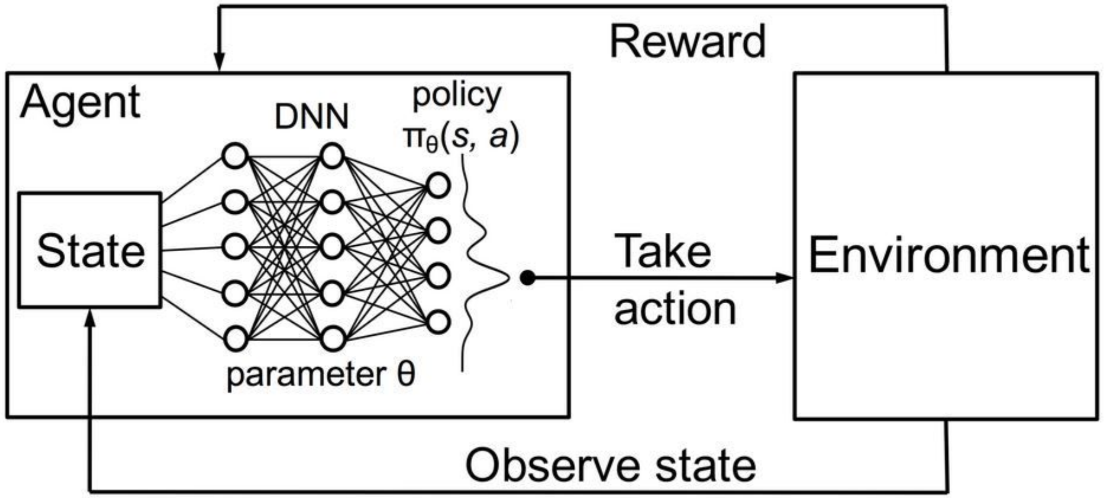
Types of RL algorithms



Neural networks and reinforcement learning

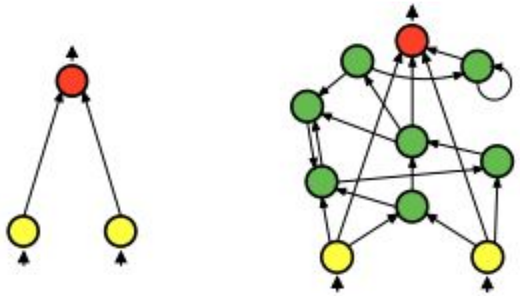
- Deep reinforcement learning





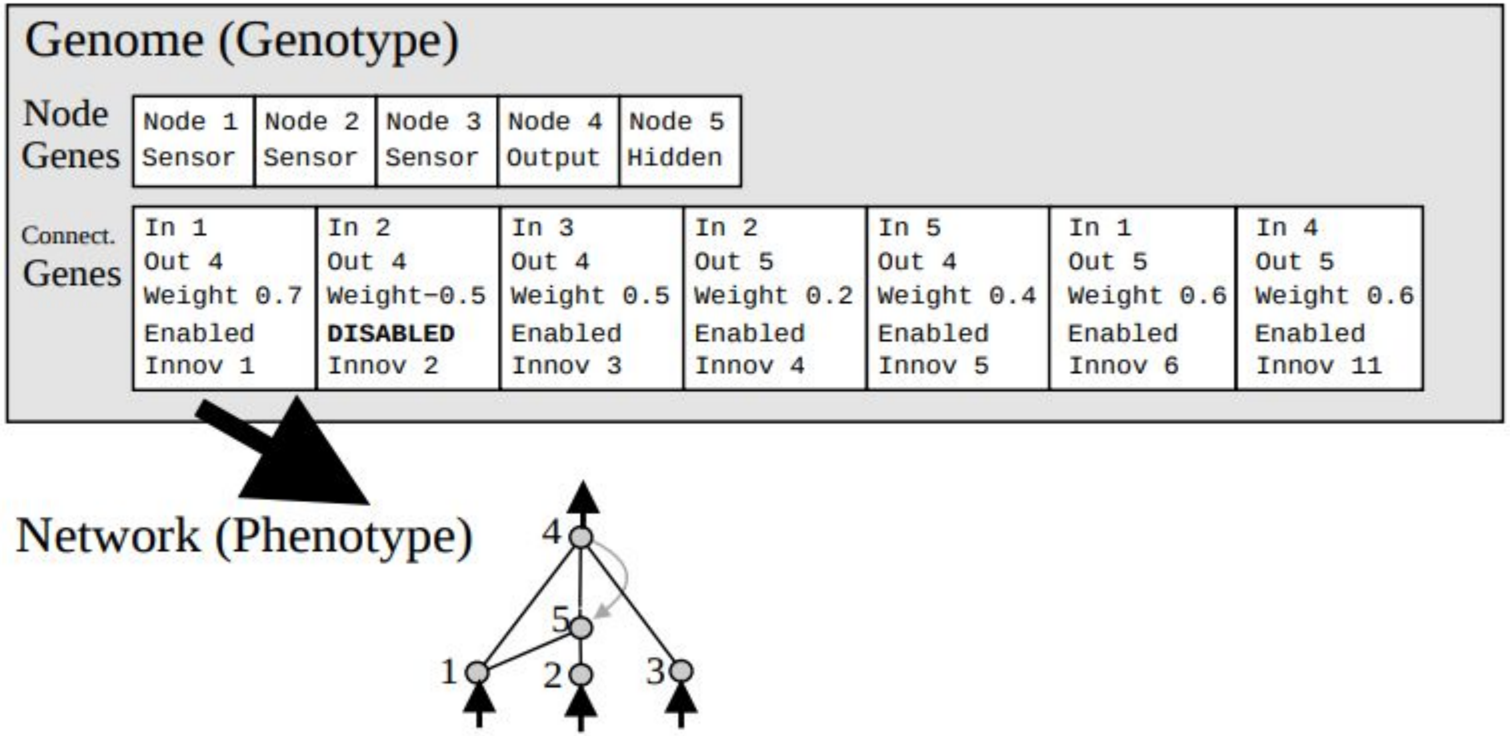
Neural networks and reinforcement learning

- NeuroEvolution of Augmenting Topologies (NEAT)
- Based on three ideas:
    - Incremental development of topologies - from simple to more complex structures
    - Using historical markings to enable crossover between matching genes
    - Using speciation in order to preserve new individuals. Enables survival of structurally innovative network. It gives them time to optimise their weights.



Neural networks and reinforcement learning

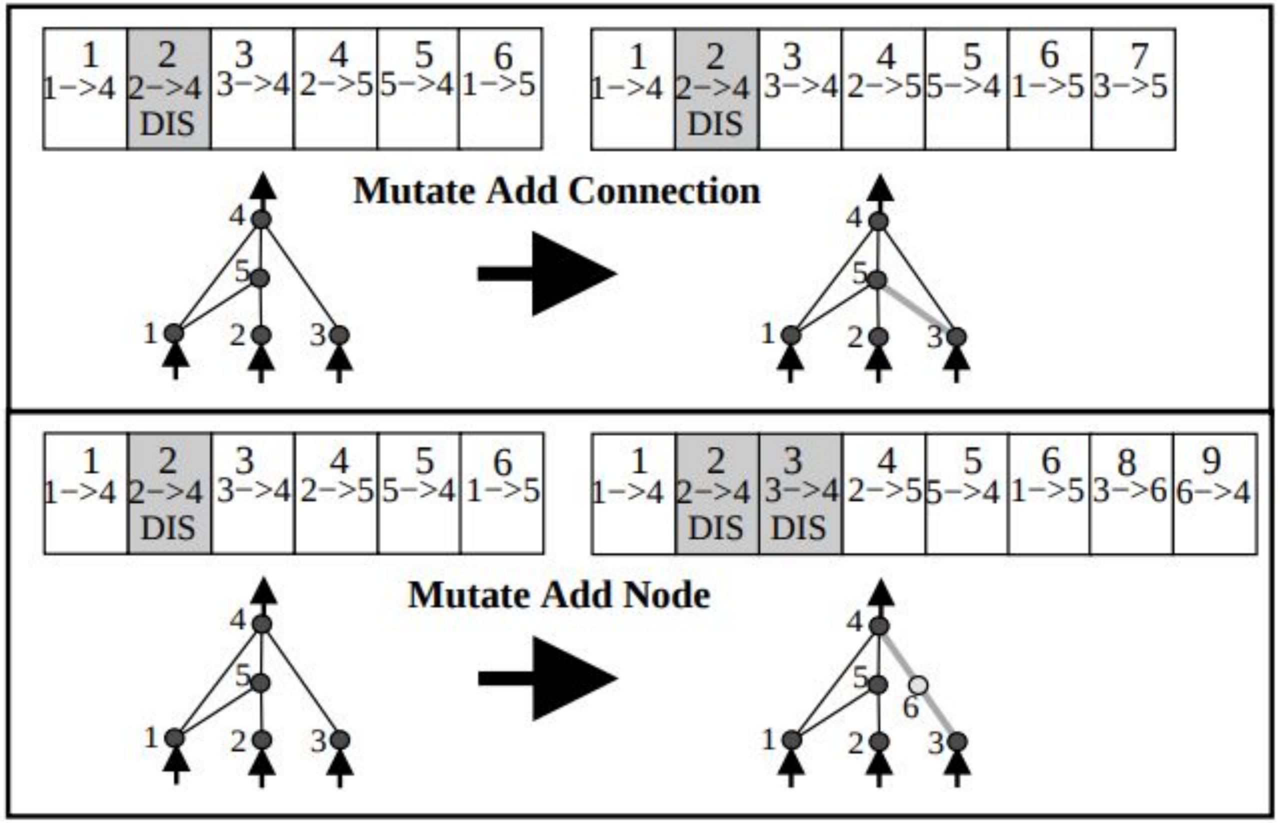
- Genom - linear representation of connections in a network. Every genome contains a list of genes
- Gene contains information on nodes and connections that connect those nodes



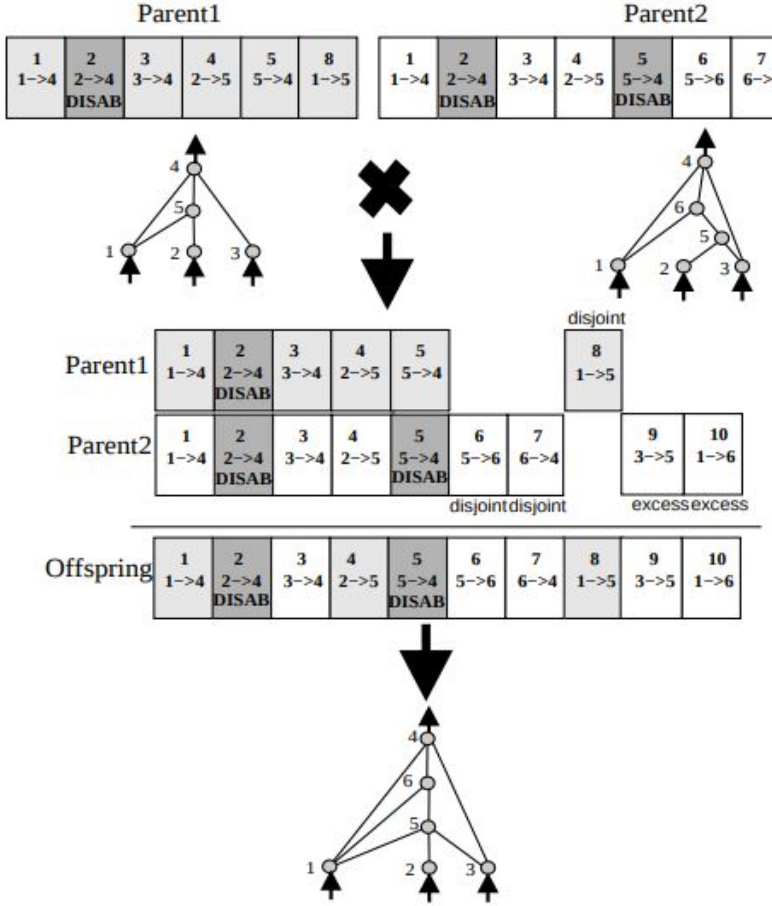
SmartCat

Neural networks and reinforcement learning

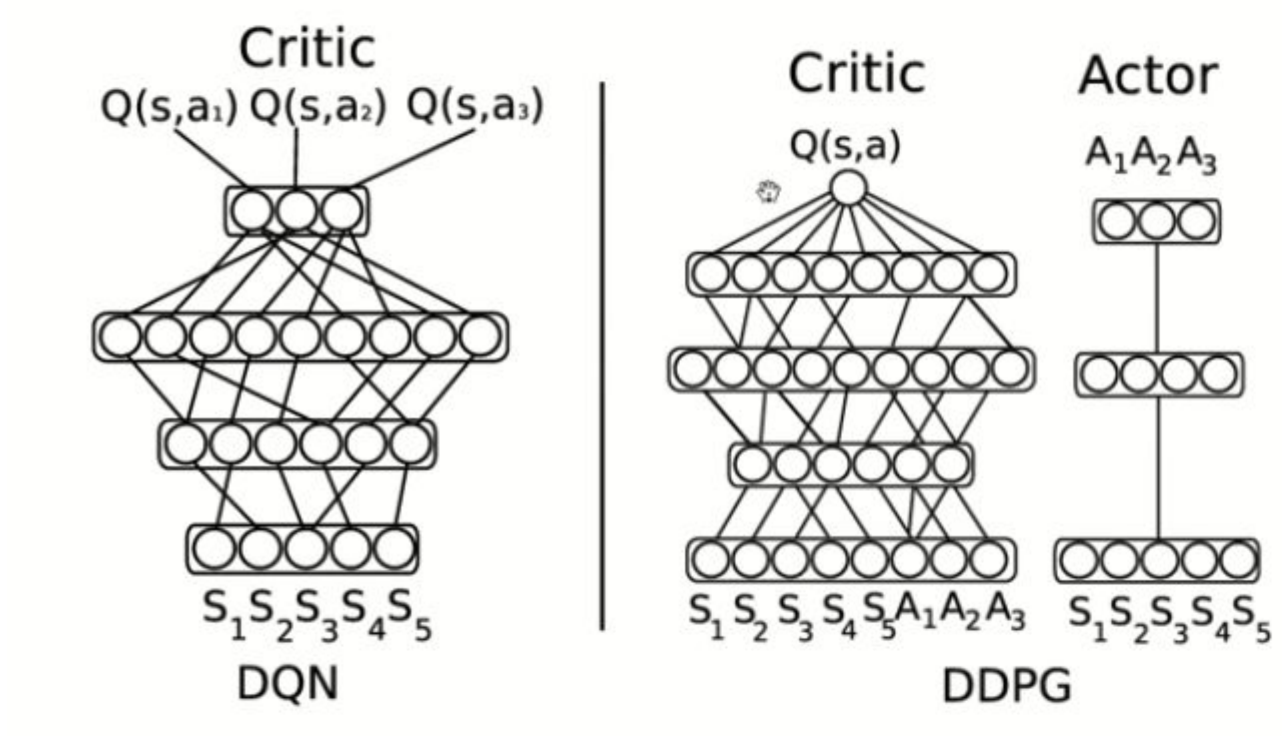
- Mutation can change both network structure (network topology) and weights



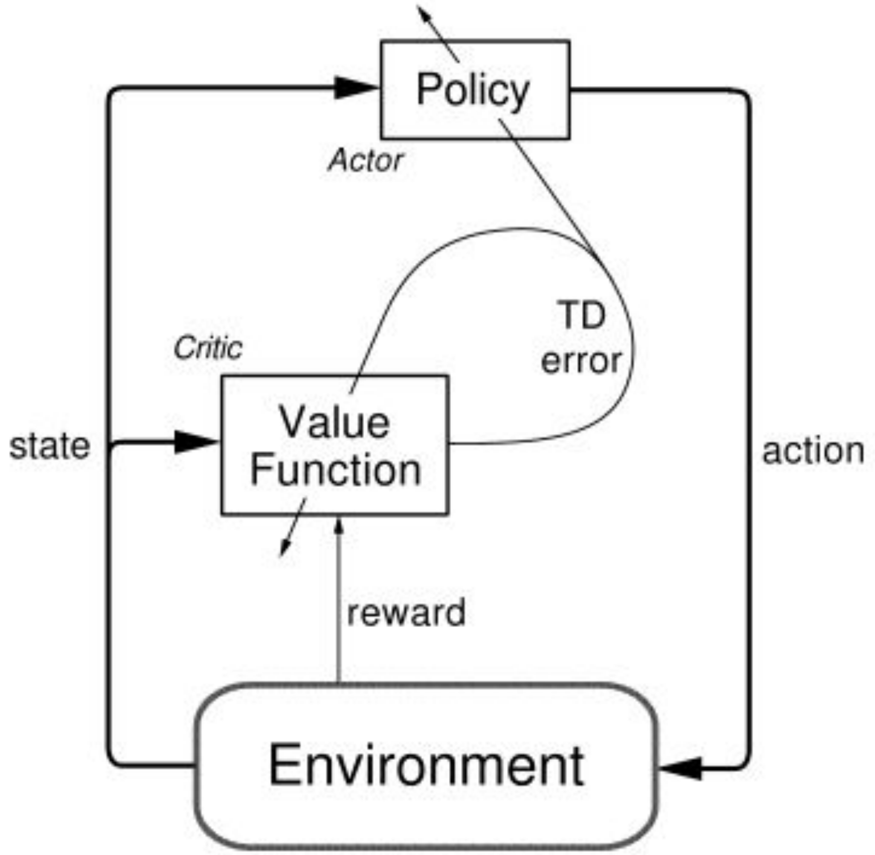
Neural networks and reinforcement learning



DDPG



Critic
Q(s,a1) Q(s,a2) Q(s,a3)
S1S2S3S4S5
DQN
Critic
Q(s,a)
S1 S2 S3 S4 S5 A1 A2 A3
Actor
A1A2A3
S1S2S3S4S5
DDPG



Policy
Actor
Critic
TD
error
state
Value
Function
action
reward
Environment

SmartCat

Workshop

Pendulum

Observation

Type: Box(3)

Num	Observation	Min	Max
0	cos(theta)	-1.0	1.0
1	sin(theta)	-1.0	1.0
2	theta dot	-8.0	8.0

Actions

Type: Box(1)

Num	Action	Min	Max
0	Joint effort	-2.0	2.0

Reward

The precise equation for reward:

-(theta^2 + 0.1*theta_dt^2 + 0.001*action^2)

Theta is normalized between -pi and pi. Therefore, the lowest cost is -(pi^2 + 0.1*8^2 + 0.001*2^2) = -16.2736044, and the highest cost is 0. In essence, the goal is to remain at zero angle (vertical), with the least rotational velocity, and the least effort.

Starting State

Random angle from -pi to pi, and random velocity between -1 and 1

Episode Termination

The episode ends when 200 iterations are reached.

Thank you

nina.marjanovic@smartcat.io
SmartCat.io